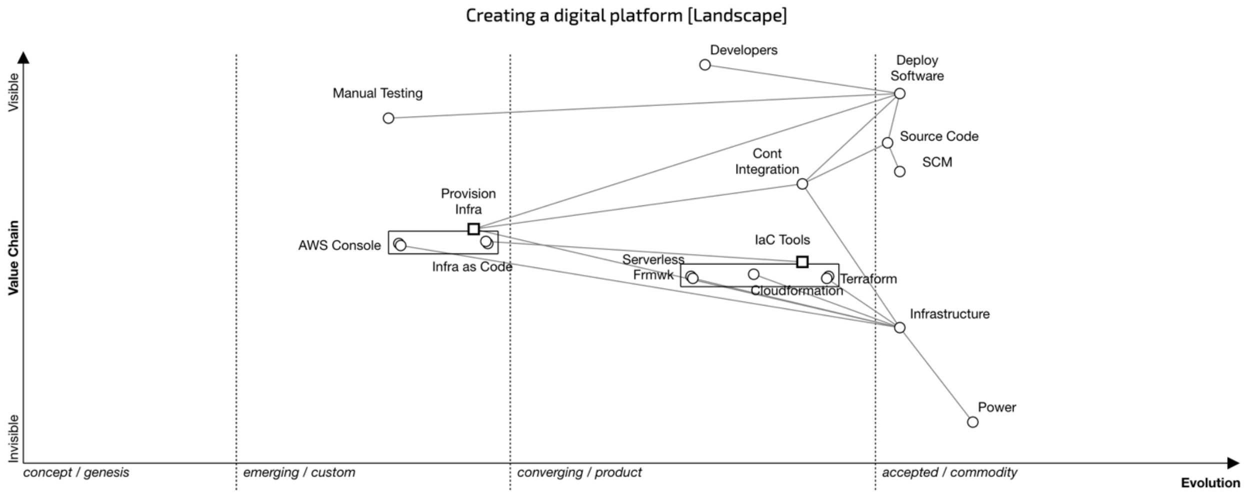
Creating a digital platform.

Here's a map of a Tech department, made up on several development teams, each with their own domain and autonomy in how things are built.

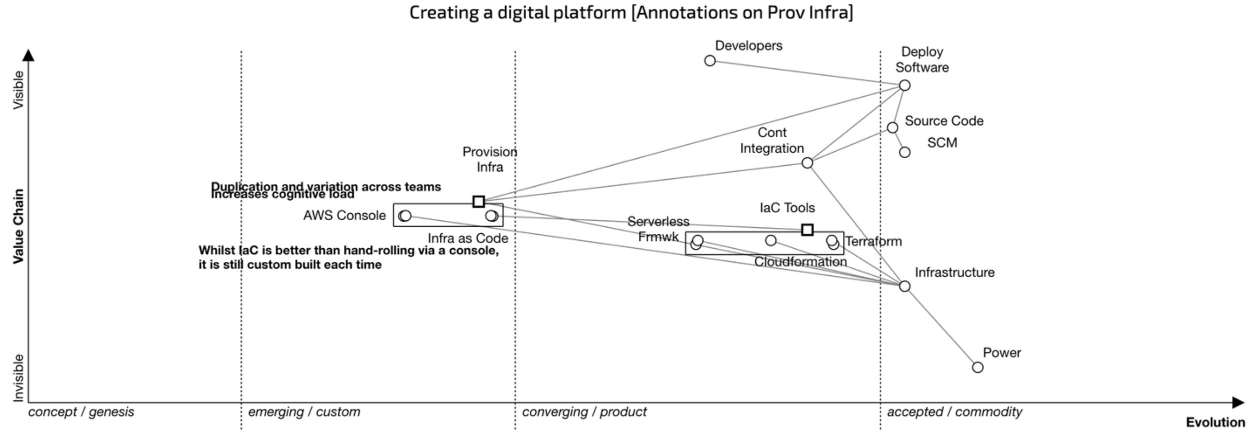
Reading through the map, Developers *need to* deploy software. To deploy software, it needs source code and to provision infrastructure to run that software. Source code is often held in SCM (source code management like GIT).



Provisioning infra.

There are many choices, developers can hand roll in the AWS Console – this creates custom built infra and since it is hand-rolled, there is no artefact or source code other than what is held in AWS. If you adopt a test environment, you might create something slightly different. Because it isn't code, we lack a guardrail to ensure consistency.

Another option is to custom-build infra as code. This is better than hand rolling infra via a console because it makes it repeatable, you can store it in source control and get all kinds of other benefits – versioning, consistency etc. But hand-cranked infra as code is still unique and custom to the application – although the test/prod environments could be the same.

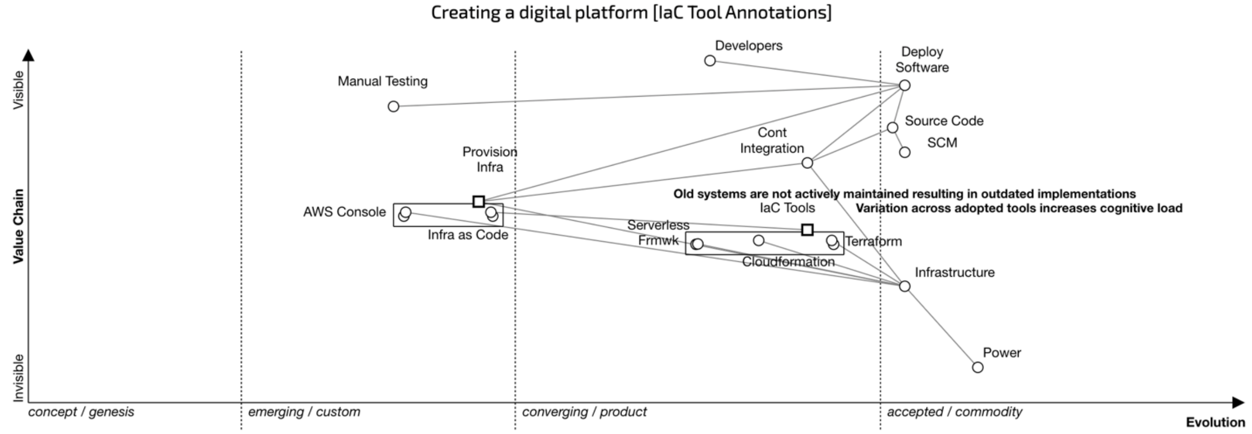


Duplication and variation.

When you scale this across teams, you get duplication and variation in approaches. You do not get economies of scale/reuse. One setup to deploy an application used in team a will/could be different in team b.   Not to mention the many different underlying tools – is it CodeFormation, Terraform or Serverless Framework. Take your pick.

Over time, this compounds future undesirable effects. An application could be running an out-of-date version of Terraform. The version is no longer supported or there are breaking changes between versions. Or even a change in security policy means you need to review the setup. You now have many instances to review and potentially update.

People move teams, adopt domains, what they see if unfamiliar. All of this contributes to cognitive load and slows us down.



Productising infrastructure

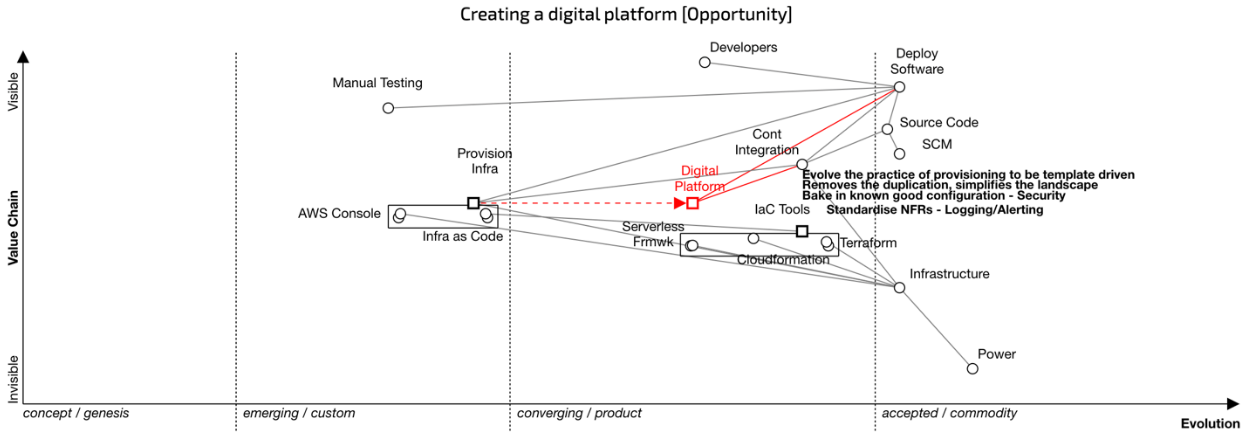
An alternative is to introduce a capability that makes your infrastructure consistent. ‘Productising’ the underlying infrastructure provisioning to be an experience like buying a laptop. Slightly different setups that you can pick/choose from depending upon your use case.

How would this look?

The different options should be drafted as infrastructure as code to create templates. These templates embody the best approaches, take care of NFRs like alerting/monitoring and security.

Traditionally we store the IaC with the application source code. What is needed here is to consider how future changes to the adopted template can be pushed out. We could take an approach of inverting the responsibility so that the application source contains configuration to a digital platform, and the digital platform assumes responsibility for pulling in the template and processing the underlying mechanism of provisioning.

Custom-building a digital platform.



In words of Simon Wardley…

> So be careful … when something is being adopted by the late majority, it can be precisely the wrong time to invest if it's on the cusp of becoming a utility. As a general rule, you want to be a fast follower to products but a first mover to utilities.

Should you custom-build a digital platform? Taking the advice above, ‘you want to be a fast follower to products, but a first move to utilities.’

Utilities represent ‘as a service’, think of what AWS has done to compute, you can rent compute (EC2) without purchasing the infrastructure. That’s commoditised form of compute. Utilities embody the learnings and represent a known/accepted approach due to their evolution.

Now what is AWS (or others) doing?

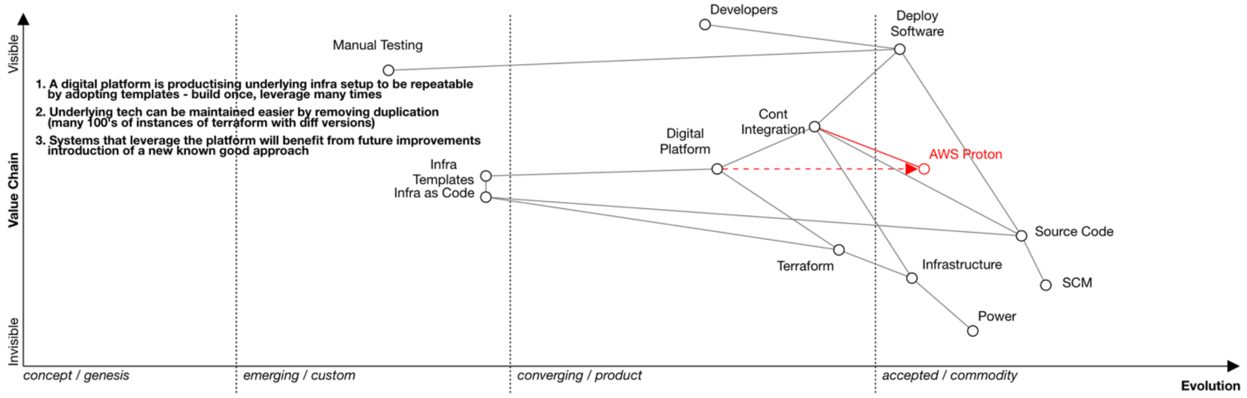
Enter AWS Proton.

Yes, AWS has created a platform to ‘give developers an easy way to deploy their code using containers and serverless technologies, using management tools, governance and visibility needed to provide consistent standards and best practice.’

And yes, it is template based.

So should you invest in building your own? Would it give you a competitive advantage?

Creating a digital platform [Future]



1. A digital platform is productising underlying infra setup to be repeatable by adopting templates - build once, leverage many times
2. Underlying tech can be maintained easier by removing duplication (many 100's of instances of terraform with diff versions)
3. Systems that leverage the platform will benefit from future improvements introduction of a new known good approach